will probably find K if k and M have been chosen properly. The chance of a false drop when there are N records in the file is approximately $(1 - e^{-kN/M})^k$. In a sense, Bloom's method treats the entire file as one record, with the primary keys as the attributes that are present, and with superimposed coding in a huge M-bit field.

Still another variation of superimposed coding has been developed by Richard A. Gustafson [Ph.D. thesis (Univ. South Carolina, 1969)]. Suppose that we have N records and that each record possesses six attributes chosen from a set of 10,000 possibilities. The records may, for example, stand for technical articles and the attributes may be keywords describing the article. Let h be a hash function that maps each attribute into a number between 0 and 15. If a record has attributes $a_1, a_2, \ldots, a_6$, Gustafson suggests mapping the record into the 16-bit number $b_0 b_1 \ldots b_{15}$, where $b_i = 1$ if and only if $h(a_j) = i$ for some j; and furthermore if this method results in only k of the b's equal to 1, for $k < 6$, another $6-k$ 1s are supplied by some random method (not necessarily depending on the record itself). There are $\binom{16}{6} = 8008$ sixteen-bit codes in which exactly six 1 bits are present, and with luck about $N/8008$ records will be mapped into each value. We can keep 8008 lists of records, directly calculating the address corresponding to $b_0 b_1 \ldots b_{15}$ using a suitable formula. In fact, if the 1s occur in positions $0 \le p_1 < p_2 < \cdots < p_6$, the function

$$\binom{p_1}{1} + \binom{p_2}{2} + \cdots + \binom{p_6}{6}$$

will convert each string $b_0 b_1 \ldots b_{15}$ into a unique number between 0 and 8007, as we have seen in exercises 1.2.6–56 and 2.2.6–7.

Now if we want to find all records having three particular attributes A_1, A_2, A_3, we compute $h(A_1)$, $h(A_2)$, $h(A_3)$; assuming that these three values are distinct, we need only look at the records stored in the $\binom{13}{3} = 286$ lists whose bit code $b_0 b_1 \ldots b_{15}$ contains 1s in those three positions. In other words, only $286/8008 \approx 3.5$ percent of the records need to be examined in the search.

See the article by C. S. Roberts, *Proc. IEEE/* **67** (1979), 1624–1642, for an excellent exposition of superimposed coding, together with an application to a large database of telephone-directory listings. An application to spelling-check software is discussed by J. K. Mullin and D. J. Margoliash, *Software Practice & Exper.* **20** (1990), 625–630.

Combinatorial hashing. The idea underlying Gustafson's method just described is to find some way to map the records into memory locations so that comparatively few locations are relevant to a particular query. But his method applies only to inclusive queries when the individual records possess few attributes. Another type of mapping, designed to handle arbitrary basic queries like (10) consisting of 0's, 1's, and $*$'s, was discovered by Ronald L. Rivest in 1971. [See *SICOMP* **5** (1976), 19–50.]

Suppose first that we wish to construct a crossword-puzzle dictionary for all six-letter words of English; a typical query asks for all words of the form `N**D*E`, say, and gets the reply {`NEEDLE`, `NIDDLE`, `NODDLE`, `NOODLE`, `NUDDLE`}. We